STIC Search Report
EIC 2100

STIC Database Tracking Number: 192185

TO: Susan Chen **Location: RND 3c25** **Art Unit: 2161** **Wednesday, June 07, 2006** **Case Serial Number: 10/056821**	**From: Emory Damron** **Location: EIC 2100** **RND 4B19** **Phone: 571-272-3520** **Emory.Damron@uspto.gov**

Search Notes

Dear Susan,

Please find below your fast and focused search results.
References of potential pertinence have been tagged, but please review all the packets in case you like something I didn't.
Of those references which have been tagged, please note any manual highlighting which I've done within the document.
In addition to searching on Dialog, I also searched JPO/Derwent, IEEE, and Inspec.
There may be a few decent references contained herein, but I'll let you determine how useful they may be to you.
Please contact me if I can refocus or expand any aspect of this case, and please take a moment to provide any feedback (on the form provided) so EIC 2100 may better serve your needs. Good Luck!

Sincerely,
Emory Damron
Technical Information Specialist
EIC 2100, US Patent & Trademark Office
Phone: (571) 272-3520
Emory.damron@uspto.gov

```
Set     Items   Description
S1        453   S CACHESERVER? OR (CACHE? OR CACHING? OR BUFFER?)()(SERVER? OR ENGINE? OR
COMPUTER? OR WORKSTATION?) OR CACHEENGINE?
S2          0   S BUFFERSERVER? OR BUFFERENGINE?
S3        453   S S1:S2
S4          5   S ONLINE? OR ON()LINE?
S5          2   S (LOG OR LOGS OR LOGGED OR LOGGING OR SIGN?)()(IN OR ON) OR LOGIN? OR
LOGON? OR SIGNIN OR SIGNINS OR SIGNON? ?
S6          2   S OFFLINE? OR OFF()LINE? OR SIGNOFF? OR LOGOFF? OR SIGNOUT? OR LOGOUT?
S7          1   S (LOGG? OR LOG? ? OR SIGN?)()(OFF OR OUT)
S8         94   S PLURAL? OR MULTIP? OR BUNCH? OR MULTIT? OR SEVERAL? OR MANY OR
ASSORTMENT?
S9        307   S NETWORK? OR WEB OR WORLDWIDEWEB?
S10       192   S ORIGIN OR INITIAL? OR MAIN OR CHIEF OR CENTRAL OR INTERNET? OR
ENTERPRISE?
S11       240   S MANAG? OR CONTROL? OR ORIGINAL? OR 1ST OR FIRST? OR INITIAL? OR SOURCE?
OR ISSUER? OR ISSUING? OR SEMINAL?
S12        45   S PRIMARY? OR MASTER? OR SUPERVIS? OR LEADER? OR HOST?
S13         6   S NUMBER()(ONE OR 1) OR PRINCIPAL? OR HEAD
S14       408   S SERVER?
S15       150   S COMPUTER? OR COMPUTING()(DEVIC? OR APPARATUS? OR SYSTEM? OR ENTIT? OR
MODUL? OR UNIT? ? OR HARDWAR? OR COMPONENT?)
S16        72   S CONVERT? OR CHANGE? OR CHANGING? OR MODIF? OR TRANSLAT? OR MODULAT? OR
ALTER?
S17        69   S TRANSFORM? OR ADAPT? OR MANIPULAT? OR AMEND? OR EDIT? OR UPDAT?
S18        28   S AUTOMAT? OR AUTO OR AUTOMATIC? OR SPONTAN? OR REFLEX?
S19         0   S AUTOGENERAT? OR AUTOTRIGGER? OR AUTOTRIGGER? OR SELFTRIGGER? OR
SELFSTART? OR AUTOSTART? OR SELFCOMMENC? OR AUTOCOMMENC? OR SELFLAUNCH? OR AUTOLAUNCH?
S20         0   S AUTOCONFIGUR? OR SELFCONFIGUR? OR (AUTO OR SELF)()CONFIGUR?
S21         9   S EXPIR? OR LAPS? OR (TIME? OR TIMING)()OUT
S22         1   S OBSOLET? OR TIME()SENSITIV? OR (TEMPORAR? OR IMPERMANEN? OR
LIMIT?)(2W)(WINDOW? OR TIMEWINDOW?)
S23        13   S INTERRUPT? OR NOGO OR NO()GO OR ABORT? OR TERMINAT? OR DISCONNECT?
S24        11   S SHUTOFF? OR SHUT?()OFF OR POWEROFF? OR POWER?()OFF OR DISABL? OR
DEACTIVAT? OR DE()ACTIVAT? OR EXPIR?
S25         2   S HALT? OR ARREST? OR CEASE? OR CEASING? OR CESSAT? OR DESIST? OR PAUS? OR
STOP?
S26        21   S DISARM? OR REMOV? OR DELET?
S27         1   S TIMESTAMP? OR DATESTAMP? OR TIMEMARK? OR DATEMARK? OR POSTMARK? OR
POST()MARK?
S28       375   S IC=G06F?
S29       305   S MC=T01?
S30       123   S S3 AND S8:S9(7N)S10:S13(7N)S14:S15
S31         0   S S30 AND S18:S20(7N)S21:S27
S32         0   S S3 AND S10:S13(7N)S14:S15 AND S18:S20(7N)S21:S27
S33         1   S S3 AND S14:S15 AND S21:S27 AND S4:S7
S34         0   S S3 AND S18:S20(7N)S21:S27
S35        17   S S3 AND S1:S2(7N)S21:S27
S36       119   S S30 AND (S4:S7 OR S16:S29)
S37       136   S S30:S36
S38        50   S S37 AND AC=US/PR
S39        20   S S38 AND AY=(1970:1999)/PR
S40        16   S S38 NOT AY=(2000:2006)/PR
S41        86   S S37 NOT S38
S42        13   S S41 AND AY=1970:1999
S43        43   S S41 NOT AY=2000:2006
S44        11   S S41 AND PY=1970:1999
S45         7   S S41 NOT PY=2000:2006
```

```
S46         67    S S39:S40 OR S42:S45
S47         67    IDPAT (sorted in duplicate/non-duplicate order)

?  show files
```

[File 347] **JAPIO** Dec 1976-2005/Dec(Updated 060404)

[File 350] **Derwent WPIX** 1963-2006/UD,UM &UP=200635

**File 350: Preview the enhanced DWPI through ONTAP DWPI (File 280). For more information, visit http://www.dialog.com/dwpi/.*

47/3,K/5 (Item 5 from file: 350) **Links**
Derwent WPIX

016443879 **Image available**
WPI Acc No: 2004-601795/200458
Related WPI Acc No: 2001-281193; 2003-800748
XRPX Acc No: N04-475790

Dynamic data object updating method for dynamic application caching, involves determining whether time-to-live period of copy of data object stored in cache server has expired, based on hit-rate, change-rate and freshness of object

Patent Assignee: PIVIA INC (PIVI-N)
Inventor: FEIERTAG M A; HOFFMAN D M; JORDAN D S; MOHAN S; TESH R M
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
US 6772203	B1	20040803	US 99312308	A	19990514	200458 B
			US 2002143041	A	20020509	

not good — now this Pat. 6505230 don't have the support for online / offline state data processing — not good

Priority Applications (No Type Date): US 2002143041 A 20020509; US 99312308 A 19990514
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 6772203	B1		20	G06F-013/00	CIP of application US 99312308 CIP of patent US 6505230

Dynamic data object updating method for dynamic application caching, involves determining whether time-to-live period of copy of data object stored in cache server has expired, based on hit-rate, change-rate and freshness of object

Abstract (Basic):

... the time-to-live period associated with a copy of data object stored in a **cache server** has **expired**, based on the hit-rate, **change**-rate and freshness of the data object. The data object is transmitted to client devices through network, only when the data object has not **expired**.

... 1) dynamic data object **updating** apparatus...

...2) dynamic data object **updating** system; and...

...3) **computer**-readable medium storing dynamic data object **updating** program...

...For **updating** dynamic data objects such as web pages for dynamic application caching for data communication between client devices such

as notebook **computer** and desktop **computer** through **networks** such as **internet**, intranet, local area **network** (LAN), wide area **network** (WAN) and virtual private **network** (VPN...

...Enables **updating** dynamic data objects reliably and easily, without increasing processing load and processing time...

...DESCRIPTION OF DRAWING - The figure shows a block diagram of the inline **server** and **offline server**.

...Title Terms: **UPDATE**;
International Patent Class (Main): **G06F-013/00**
Manual Codes (EPI/S-X): **T01-N01D2**...

...**T01-N02B1A**...

...**T01-S03**

47/3,K/25 (Item 25 from file: 350) **Links**
Derwent WPIX

012756113 **Image available**
WPI Acc No: 1999-562230/199947
XRPX Acc No: N99-415365
Updated document distribution method used in computer networks used in electronic commerce etc.,
Patent Assignee: INFOLIBRIA INC (INFO-N)
Inventor: HEDDAYA A A; MIRDAD S A; YATES D J; YATES I C
Number of Countries: 086 Number of Patents: 005
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
WO 9948003	A2	19990923	WO 99US4738	A	19990303	199947	B
ZA 9902138	A	19991229	ZA 992138	A	19990317	200006	
AU 9928937	A	19991011	AU 9928937	A	19990303	200008	
EP 1076978	A2	20010221	EP 99909815	A	19990303	200111	
			WO 99US4738	A	19990303		
US 6205481	B1	20010320	US 9840520	A	19980317	200118	

CLAIMS 2+12

Priority Applications (No Type Date): US 9840520 A 19980317
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
WO 9948003	A2	E	53	G06F-012/00	

Designated States (National): AL AM AT AU AZ BA BB BG BR BY CA CH CN CU CZ DE DK EE ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MD MG MK MN MW MX NO NZ PL PT RO RU SD SE SG SI SK SL TJ TM TR TT UA UG UZ VN YU ZW
Designated States (Regional): AT BE CH CY DE DK EA ES FI FR GB GH GM GR IE IT KE LS LU MC MW NL OA PT SD SE SL SZ UG ZW

ZA 9902138	A		53	H04L-000/00	
AU 9928937	A			G06F-012/00	Based on patent WO 9948003
EP 1076978	A2	E		H04L-029/08	Based on patent WO 9948003

Designated States (Regional): AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE

US 6205481	B1			G06F-015/173	

Updated document distribution method used in computer networks used in electronic commerce etc.,

Abstract (Basic):
... The request output by a client is received by **cache servers** (16) which are distributed between home server (20- 1) and clients (12-1 - 12-4...

...outputs local cache copy corresponding to the output request. The cache copy stored by neighboring **cache server**, is stored and **updated** after determining identity of neighboring **cache server**.

... For distributing **updated** documents between clients connected through **computer networks** like **internet**, private intranet, extranet, virtual private **networks**. Utilized in retrieval of information, communication, electronic commerce, entertainment and other applications and for sharing...

...Upon sending a message to neighboring **cache server**, a request is made to return the requested document copy, if more recent copy contained in neighboring, cache, which preferably cooperate to ensure documents list remain **updated**, so that rate of queries submitted to home servers is reduced. Avoids need for clients...

...shows the typical computer network showing request path for a single document and location of **cache servers**.

Title Terms: **UPDATE**;
International Patent Class (Main): **G06F-012/00**...

...**G06F-015/173**
International Patent Class (Additional): **G06F-015/167**...

...**G06F-017/30**
Manual Codes (EPI/S-X): **T01-H**

47/3,K/4 (Item 4 from file: 350) **Links**
Derwent WPIX

016695878 **Image available**
WPI Acc No: 2005-020157/200502
Related WPI Acc No: 2004-477678
XRPX Acc No: N05-017134

Method of caching network file on personal computer, involves writing data into cache, server file or to both cache and server file, based on online and off line state of computer and property set of server file

Patent Assignee: MICROSOFT CORP (MICT)
Inventor: CORRINGTON R E; LINN J L; PARDIKAR S; RAMAN B S
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
US 20040236777	A1	20041125	US 98134720	A	19980814	200502 B
			US 2004880056	A	20040630	

Priority Applications (No Type Date): US 98134720 A 19980814; US 2004880056 A 20040630
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 20040236777	A1		18	G06F-007/00	Cont of application US 98134720 Cont of patent US 6757705

Method of caching network file on personal computer, involves writing data into cache, server file or to both cache and server file, based on online and off line state of computer and property set of server file

Abstract (Basic):
... The **online** and **off line** state of the local personal computer and the property set of the server file accessible...
... For caching network files for **off line** use on personal computer, handheld device, multiprocessor system, microprocessor-based system or programmable consumer electronics, minicomputer and mainframe **computer** connected to **network** such as local area **network** (LAN), wide area **network** (WAN), intranet, **internet**, **enterprise**-wide **computer network**, in computing environment...

...Enables caching suitable network files efficiently and transparently. Enables precise **online** path and the filename to be recreated by the caching mechanism in **off line**. Improves the network and file server performance even when several users access many files concurrently...

International Patent Class (Main): **G06F-007/00**
Manual Codes (EPI/S-X): **T01-N01D4**...

...T01-N02A2C...

...T01-S03

47/3,K/31 (Item 31 from file: 350) **Links**
Derwent WPIX

012094441 **Image available**
WPI Acc No: 1998-511352/**199844**
XRPX Acc No: N98-399065
Proxy cache server control method - involves deleting data of upper stage proxy cache server when it is checked that all lower stage proxy cache servers positioned directly under it perform cache of certain identical data

Patent Assignee: NIPPON TELEGRAPH & TELEPHONE CORP (NITE)
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
JP 10222411	A	19980821	JP 9723013	A	19970205	199844	B

Priority Applications (No Type Date): JP 9723013 A 19970205
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
JP 10222411	A		6	G06F-012/00	

Proxy cache server control method...
...**involves deleting data of upper stage proxy cache server when it is checked that all lower stage proxy cache servers positioned directly under it perform cache of certain identical data**

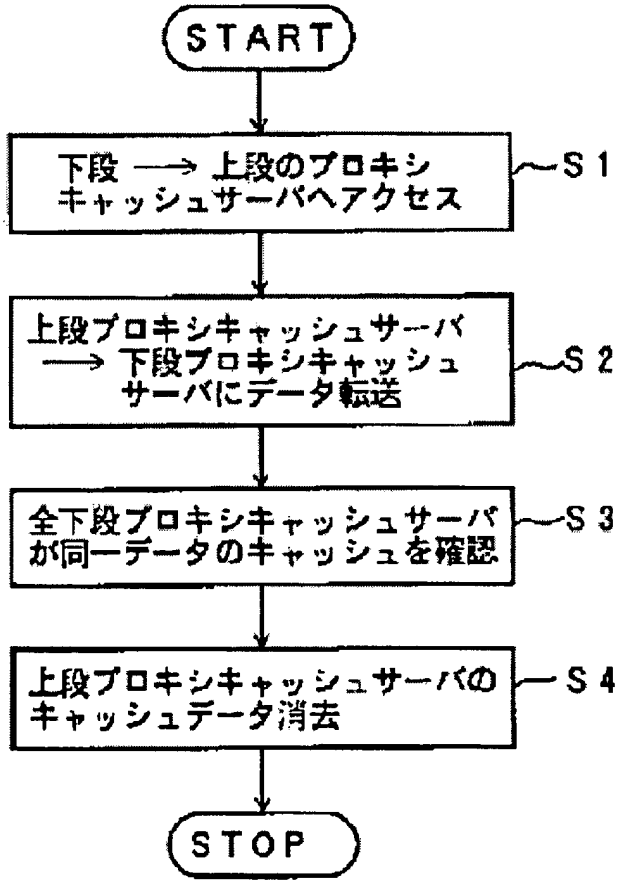
...Abstract (Basic): The method involves **deleting** data of an upper stage proxy **cache server** (100) among multi-stage proxy **cache server**. The data is **deleted** when it is checked that all the lower stage proxy **cache servers** (200) which are positioned directly under the upper stage. Proxy **cache server**, performs cache of certain identical data...

PROXY CACHE SERVER CONTROL METHOD AND PROXY CACHE SERVER

Patent number: JP10222411
Publication date: 1998-08-21
Inventor: ASAKA TAKUYA
Applicant: NIPPON TELEGRAPH & TELEPHONE
Classification:
- international: ***G06F12/00;*** **G06F12/00;** (IPC1-7): G06F12/00
- european:
Application number: JP19970023013 19970205
Priority number(s): JP19970023013 19970205

Abstract of **JP10222411**

PROBLEM TO BE SOLVED: To leave data which is possibly referred to in an upper stage proxy cache server, to effectively use cache possible data capacity and to improve the hit ratio of cache data by erasing data of the upper stage proxy cache server when all lower stage proxy cache servers positioned immediately under the upper proxy cache server are recognized to cache same data. SOLUTION: The lower stage proxy cache server connected to a user host accesses to the upper stage proxy cache server among the multistage proxy cache servers (step 1). Cache data is transferred from the upper stage cache server to the lower stage proxy cache server (step 2). When the upper stage proxy cache server recognizes that all the lower stage proxy cache servers cache same data (step 3), it erases cache data (step 4).



```
Set     Items   Description
S1        733   S CACHESERVER? OR (CACHE? OR CACHING? OR BUFFER?)()(SERVER? OR ENGINE? OR
COMPUTER? OR WORKSTATION?) OR CACHEENGINE?
S2          0   S BUFFERSERVER? OR BUFFERENGINE?
S3        733   S S1:S2
S4         17   S ONLINE? OR ON()LINE? OR NETWORK?(2N)CONNECTED
S5          1   S (LOG OR LOGS OR LOGGED OR LOGGING OR SIGN?)()(IN OR ON) OR LOGIN? OR
LOGON? OR SIGNIN OR SIGNINS OR SIGNON? ?
S6          2   S OFFLINE? OR OFF()LINE? OR SIGNOFF? OR LOGOFF? OR SIGNOUT? OR LOGOUT?
S7          0   S (LOGG? OR LOG? ? OR SIGN?)()(OFF OR OUT)
S8        210   S PLURAL? OR MULTIP? OR BUNCH? OR MULTIT? OR SEVERAL? OR MANY OR
ASSORTMENT?
S9        555   S NETWORK? OR WEB OR WORLDWIDEWEB?
S10       371   S ORIGIN OR INITIAL? OR MAIN OR CHIEF OR CENTRAL OR INTERNET? OR
ENTERPRISE?
S11       363   S MANAG? OR CONTROL? OR ORIGINAL? OR 1ST OR FIRST? OR INITIAL? OR SOURCE?
OR ISSUER? OR ISSUING? OR SEMINAL?
S12        41   S PRIMARY? OR MASTER? OR SUPERVIS? OR LEADER? OR HOST?
S13         7   S NUMBER()(ONE OR 1) OR PRINCIPAL? OR HEAD
S14       572   S SERVER?
S15       391   S COMPUTER? OR COMPUTING()(DEVIC? OR APPARATUS? OR SYSTEM? OR ENTIT? OR
MODUL? OR UNIT? ? OR HARDWAR? OR COMPONENT?)
S16       110   S CONVERT? OR CHANGE? OR CHANGING? OR MODIF? OR TRANSLAT? OR MODULAT? OR
ALTER?
S17       121   S TRANSFORM? OR ADAPT? OR MANIPULAT? OR AMEND? OR EDIT? OR UPDAT?
S18        43   S AUTOMAT? OR AUTO OR AUTOMATIC? OR SPONTAN? OR REFLEX?
S19         0   S AUTOGENERAT? OR AUTOTRIGGER? OR AUTOTRIGGER? OR SELFTRIGGER? OR
SELFSTART? OR AUTOSTART? OR SELFCOMMENC? OR AUTOCOMMENC? OR SELFLAUNCH? OR AUTOLAUNCH?
S20         0   S AUTOCONFIGUR? OR SELFCONFIGUR? OR (AUTO OR SELF)()CONFIGUR?
S21         7   S EXPIR? OR LAPS? OR (TIME? OR TIMING)()OUT
S22         0   S OBSOLET? OR TIME()SENSITIV? OR (TEMPORAR? OR IMPERMANEN? OR
LIMIT?)(2W)(WINDOW? OR TIMEWINDOW?)
S23         7   S INTERRUPT? OR NOGO OR NO()GO OR ABORT? OR TERMINAT? OR DISCONNECT?
S24         7   S SHUTOFF? OR SHUT?()OFF OR POWEROFF? OR POWER?()OFF OR DISABL? OR
DEACTIVAT? OR DE()ACTIVAT? OR EXPIR?
S25         3   S HALT? OR ARREST? OR CEASE? OR CEASING? OR CESSAT? OR DESIST? OR PAUS? OR
STOP?
S26        23   S DISARM? OR REMOV? OR DELET?
S27         1   S TIMESTAMP? OR DATESTAMP? OR TIMEMARK? OR DATEMARK? OR POSTMARK? OR
POST()MARK?
S28        41   S S3 AND S21:S27
S29        41   S S28 AND S4:S20
S30        41   S S28:S29
S31        15   S S30 AND PY=1970:1999
S32        17   S S30 NOT PY=2000:2006
S33        17   S S31:S32
S34        16   RD  (unique items)
 ; show files
```

[File 2] **INSPEC** 1898-2006/May W4

[File 6] **NTIS** 1964-2006/May W3

[File 8] **Ei Compendex(R)** 1970-2006/May W4

NONPAT
LIT
BIBLIO
FILES

[File 34] **SciSearch(R) Cited Ref Sci** 1990-2006/May W4

[File 35] **Dissertation Abs Online** 1861-2006/May

[File 56] **Computer and Information Systems Abstracts** 1966-2006/May

[File 60] **ANTE: Abstracts in New Tech & Engineer** 1966-2006/May

[File 65] **Inside Conferences** 1993-2006/Jun 06

[File 94] **JICST-EPlus** 1985-2006/Mar W1

[File 99] **Wilson Appl. Sci & Tech Abs** 1983-2006/Apr

[File 111] **TGG Natl.Newspaper Index(SM)** 1979-2006/May 29

[File 144] **Pascal** 1973-2006/May W2

[File 239] **Mathsci** 1940-2006/Jul

[File 256] **TecInfoSource** 82-2006/Jul

34/3,K/3 (Item 3 from file: 2) **Links**
INSPEC

07040907 **INSPEC Abstract Number:** C9811-6130D-009
Title: Dealing with one-timer-documents in Web caching
Author Belloum, A.; Hertzberger, L.O.
Author Affiliation: Dept. of Comput. Sci., Amsterdam Univ., Netherlands
Conference Title: Proceedings. 24th EUROMICRO Conference (Cat. No.98EX204) **Part** vol.2 p. 544-50 vol.2
Publisher: IEEE Comput. Soc , Los Alamitos, CA, USA
Publication Date: 1998 **Country of Publication:** USA 2 vol. liv+1075 pp.
ISBN: 0 8186 8646 4 **Material Identity Number:** XX98-02494
U.S. Copyright Clearance Center Code: 1089-6503/98/$10.00
Conference Title: Proceedings 24th EUROMICRO Conference
Conference Sponsor: Sun Microsyst.; ENATOR; ABB Network Partner; Ericsson; ABB Generation; K K Stiftelsen; ABB Ind. Syst.; Malardalens Hogskola
Conference Date: 25-27 Aug. 1998 **Conference Location:** Vasteras, Sweden
Language: English
Subfile: C

Title: Dealing with one-timer-documents in Web caching
Abstract: A proper initialization requires starting the process in a state close to the expected steady state. In Web caching, the initialization problem is faced each time a new document enters the cache. Independently of the method... ...the documents into the cache, the newly referenced document is inserted in a so called "removal-list", from which documents are removed when storage space is needed. Often, undesirable documents are assigned a high priority, consequently these documents remain for quite a long time in the cache, leading to a decrease in cache server performances. We investigate one category of undesirable documents, which pass the filters commonly used to control the cache processing.
Descriptors: ...Internet; ... ...storage management
Identifiers: ...Web caching... ...initialization; ... ...removal-list... ...cache server performances
1998

Dealing with One-Timer-Documents in Web Caching

A. Belloum and L.O. Hertzberger
Computer Architecture and Parallel Systems Group
Dept. of Computer Science
Universiteit van Amsterdam
Kruislaan 403 NL 1098 SJ Amsterdam
{adam,bob}@wins.uva.nl

Abstract

A proper initialization requires starting the process in a state close to the expected steady-state. In web caching, the initialization problem is faced each time a new document enters the cache, independently of the method used to sort the documents into the cache, the newly referenced document is inserted in a so called "removal-list", from which documents are removed when storage space is needed. Often, undesirable documents are being assigned a high priority, consequently these documents remain for quite a long time in the cache, leading to a decrease in cache server performances. In this paper, we shall investigate one category of undesirable documents, which passe the filters commonly used to control the cache processing.

1. Introduction

Web caching has been introduced to solve the problem of the rapid increase of the Internet traffic. Basically, web caching consists of providing throughout the Internet web sites that keep copies of documents requested by the users. The expectation is the multiple accesses to the same documents are serviced to the user without having to go through the international busy connection to the origin server. To be efficient, the web cache servers have to be spread out over the Internet so that the end-user can redirect its request to the nearest one. This issue outlines the problem of the geographical distribution of the cache servers. Besides that, web caching mechanisms should keep in their local storage only the most frequently requested documents, those documents that generate the major part of the Internet traffic.

Web traffic analysis showed that access patterns for each document over the Internet is not uniform, the dynamics of Web traffic seem to be difficult to characterize and there are several differences between the Web and other network traffics [2, 10, 1] . This means that some sites, and more precisely, some documents at some particular sites are requested more frequently than others, and these sites and documents are spread non uniformly over the Internet [8]. Among all these requests only a small fraction accounts for most of the Internet accesses. Besides, these studies showed that the most popular files are the less frequently updated [7]. Another important characteristic of the so called popular documents is their relatively small size (less than 10 kB [4]). Therefore, it becomes clear that this category of documents is well suited for caching approaches. A large number of these documents could be cached and remain up-to-date for relatively long periods of time (usually a few days), which will lead to a dramatic reduction in both of the Internet traffic and the user perceived time[1].

Web caching would be the ideal solution to the increase of the Internet traffic if the documents would be cached for a long period of time. Unfortunately, due to the lack of storage space, the cache manager has to remove a number of cached document in order to make room for a newly referenced one.

The documents replacement strategy is composed of two phases. First, the documents are sorted in the cache in order to determine the removal ordering. This task is performed at each new request. Second, one or more documents are removed from the head of the removal list. Several strategies have been proposed such as the least frequently used (LFU), first in first out (FIFO) and many others. Only few of them have been implemented and used in real cache servers. Most of the time, documents are sorted in the cache according to the least recently used (LRU) strategy.

Our main interest in this paper is the removal list used in almost all of the document removal policies. The cached documents are stored within this list according to the different removal keys described previously. Unfortunately most of the removal policies do not have a mechanism to identify the so called *one-timer* documents. Those docu-

[1] The perceived time is the elapse time between sending the request and receiving the document

1089-6503/98 $10.00 © 1998 IEEE

ments which are requested only once or a small number of times. Often, document replacement strategies assign high priorities to *one-timer* documents allowing them to remain in the cache for quite a long period of time without being referenced again. This could considerably affect the cache performance especially when the number of *one-timer* documents is large. The web traffic analysis presented in [2] shows that about one-third of the incoming requests belongs to this category of documents. It is thus of great importance to introduce policies that could better discriminate these *one-timer* documents. One solution, proposed by the VT-NRG Group and Arlitt et al, is to keep a list of all documents that have been accessed and only retains documents in the cache when they have been requested a second time. Only the documents that are accessed twice would be cached, this approach is known as the *Ignore First Hit*. This method introduce a extra latency on the second request to the document and computational overhead due to maintaining the list of *one-timer* documents.

The rest of this paper is organized as follows: in Section 2, a number of removal policies are presented and the impact of the *the one-timer documents* on these policies is discussed. In Section 3, a new method is proposed to deal with the problem of *one-timer documents*. In Section 4, a set of experiments are presented identifying the impact the proposed method to deal with *one-timer documents*. Finally Section 5 concludes the paper.

2. Replacement strategies

2.1. The LFU strategy

At first sight, the LFU (Least Frequently Used) strategy fits better to the web cache problem, which states that the most frequently requested documents are responsible for the major part of the web traffic. By maintaining a reference count for each cached document, the LFU strategy could estimate the frequency of references for each document. Two main problems arise with such an approach. First, once the cache reach its steady state and documents start being replaced frequently, the newly referenced documents are always removed first since they have the lowest reference count. The new documents have no time to get their count increasing in order to be cached for a longer period of time. Second, documents that build up an extremely high reference counts are rarely (if ever) replaced, even if they are no more requested.

The LFU strategy only focuses on one parameter in sorting the documents in the cache, it is clear that the two problems described here result form the fact that the time is ignored by the LFU strategy. To overcome these problems new versions of the LFU strategy have been proposed, the LFU_Aging and the LFU*. These studies were proposed by Arlitt in [2] to deal with respectively the high reference count and the *one-timer* documents.

The LFU_Aging avoid the building up of a high reference count by limiting and aging reference counts. To combat the high reference count, the *LFU_Aging* fixes the maximal number of reference per document and records the age of each reference count.

The LFU* differs from the LFU policy in that on a cache miss, the newly requested document is not always added to the cache. More specifically, only documents with reference count equal to one are candidate for replacement. The one timer documents are removed first but the building up of high reference count is always possible.

As stated above the LFU strategy implicitly solves the problem of *one-timer documents*. However, it could also lead to a stale cache, where the previously popular documents that have a very high reference count, will block the access to the cache for new documents getting more and more popularity.

2.2. Other replacement strategies

Except for the LFU strategy, the rest of the web caching replacement policies have no real mechanism to identify the *one-timer* documents.

Ignore the first hit is the only way to deal with *one-timer* documents. On the first request the document is not cached, only its header is kept in the cache. On the next request, the document is identified as not a *one-timer*. This technique introduce a substantial latency due to a real data transfer form the origin server on each document second hit.

Since it is not possible to know a priori if the new reference is a *one-timer* document or not. Thus, it is important that each new reference to be kept for a period of time, to make sure it will not receive other references in the nearest future. If documents identified at current time as *one-timer* are not removed first, a newly requested document is more likely to see it reference count increasing. The LRU strategy use such a mechanism, the documents are sorted according to their last reference, therefore, the least recently used document is removed first. The LRU strategy insert each new reference at the bottom of the removal list, giving it the opportunity to stay in the cache for a longer period of time. When real *one-timer* enters the cache, the LRU strategy often removes a number of multi-referenced documents, which reduces the server performances. The document replacement strategies not considering neither the time

nor the number of references, such as the SIZE based one, are the most sensitive to the *one-timer* documents. Once a *one-timer* document enters the cache it can remain for a long time (if ever).

In the previous subsections, two factors have been identified as having a direct impact on the *one-timer* document identification process: the number of references and the time of the last reference. In the following section, a method that considers both of these two factors is proposed to deal with *one-timer* document identification.

2.3. Sorting process initialization

We propose a proper initialization of the document sorting process involved in the replacement policies to deal with problem of *one-timer* documents. Assigning an initial value to each new reference in the cache will prevent from considering these documents as highly potential to be kept in the cache. The cached documents are assigned a different value only if they are requested more than once. The *one-timer* documents remain with their initial value which push them to the head of the removal list. The advantage of the sorting process initialization over the Ignore First Hit approach is for the documents requested more than once, we do not have to request again the document from the original server. Besides that, the computational overhead requires to maintain and search within the *one-timer* document list is avoided.

2.4. Sorting one-timer documents

According to the sorting process initialization method the *one-timer* documents are assigned the same initial value to distinguish them from the rest of the documents (This initial value is referenced as the *unknown priority*). Such a method of assigning priority does not allow the identification of documents within the *one-timer* category. It is therefore necessary to use a second sorting key to perform this selection process, any one of the commonly used replacement strategy could be used, however it is important to note that each *one-timer* document is likely to get a higher priority at each new request. Each *one-timer* document should remain in the cache for a certain period of time to get enough time to see whether or not its priority changed. The parameter time plays an important role in this process. Therefore the LRU based methods will fit better to this problem. Combining the initialization approach with the LRU strategy provide a simpler solution to the cache "purging" rather than the method used in cache Harvest ver1.4pl3 [2], which consists of a complicate mechanism using an upper and lower threshold to control the cache disk usage and which leads to suboptimal performance since the cache is kept below 100% utilization [10].

[2]Harvest is a proxy cache server http://harvest.cs.colorado.edu

2.5. Cache partitioning

To illustrate the initialization of the document sorting process, the cache has been partitioned into two parts: one for the *one-timer* documents (partition A) and one for the rest (partition B). In a mono-partition cache, the number of *one-timer* documents will decrease with the number of requests, since a newly referenced document will be removed first. After a while their number has become so small that any newly referenced document is likely to be removed on the next request. In such a situation, no more documents will have the chance to get higher priority, this will result in a reduction of cache performance since its content will become out-of-date after a certain time. Splitting the cache in two partitions will keep a fixed number of *one-timer* documents cached so that the updating process of the cache is always fair. The partitioning of the cache will be performed according to the total number of documents in the cache and not according to the data size. The *one-timer* documents partition is always a fixed percentage of the current number of documents. To support such a partitioning process a mechanism converts the document with the lowest priority within the partition B into a *one-timer* document as shown in Figure 1. When a document passes from partition B to partition A its priority is assigned an unknown value, which make it a *one-timer* document, since the sorting process in partition A is based on the time, this document will be pushed at the top of the removal list.

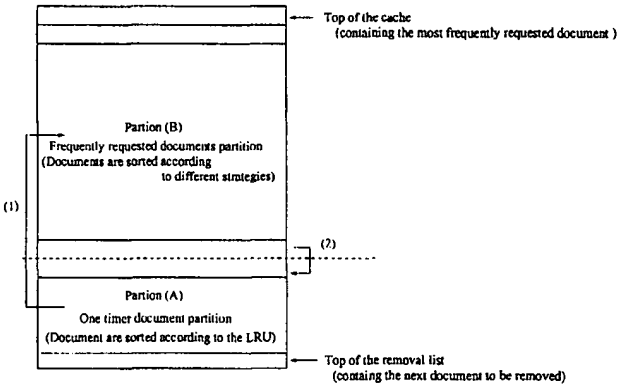


Figure 1. Cache partitioning process

3. The experiments

In the following experiments we outlined the impact of the initialization of the document sorting process on the web

cache performance by just applying this concept to the replacement strategies discussed in the previous section. The performance metrics used in this web cache analysis mainly focus on the document hit rate and the byte hit rate. The reason for this choice is the fact that byte hit rates give more information of the network bandwidth while the document hit rate focuses only on the number of requests satisfied using the cached copies. In web caching documents have different sizes, recording then only the document hit rate will not give any ideas on how the method impacts the network bandwidth. Such an information is very important for a web cache performance analysis, since the current caching methods seem to give advantage to small size document. Besides this, since we are investigating a two level cache server topology, we have redefined both the document hit rate and the byte hit rate in order to record hits on the two cache levels. The Two metrics are:

- The document hit rate "SHR": This metric records the global hit rate obtained using a two level cache configuration.
- The byte hit rate "SBHR": This metric records the global byte hit rate obtained using a two level cache configuration.

3.1. Workload traces

The workload used in this study are part of access log-files provided by the web server of the Computer Science Department of the University of Amsterdam WINS (wins.uva.nl) and the proxy server NLANR (ircache.nlanr.net). These workload traces represent two types on web cache server, more information on these workloads are provided in table 1. The WINS workload contains external requests to documents provided by the wins.uva.nl server, this type of workload trace exhibit a strong document reference locality. The NLANR workload involves requests coming form different web severs that use NLANR as a proxy server, usually this kind of servers is much more busy than the normal cache server. They receive requests for documents belonging to other web servers because usually proxy cache servers keep a copy of the documents serviced in the past. Most of the time these access log-files are large and it is very hard to use more than few days of workload duration. In the following experiments, we use these two workloads to outline the impact of the workload characteristics on the mechanism we are introducing in this paper.

3.2. The LFU strategy

In this experiment, the document replacement policy examined is the LFU. The simulation results presented in Figures 2 and 3 show a slight improvement for both of the cache

workload	Duration	Transfered data	Number of requests
WINS	1 month	4.2 GB	737750
NLANR	1 day	2.5 GB	261135
Document size	< 10 KB	< 1MB	> 1 MB
WINS	70% (1.04 GB)	9% (3.10 GB)	0.01% (0.15 GB)
NLANR	56.5% (0.47 GB)	18.4% (0.42 GB)	0.09% (0.58 GB)

Table 1. *Workloads characteristics*

metrics (SHR and SBHR) when introducing the sorting process initialization to the LFU strategy (LFU_init). The small increase in the cache performances is not only due to the small amount of *one-timer* documents involved in the WINS workload, but it is also the result of the fact that implicitly, the LFU replacement strategy has solved the problem on the *one-timer* documents by assigning a reference count to each document. Since the sorting process initialization has been introduced to deal with this problem, its impact on the cache performance have been thus considerably reduced.

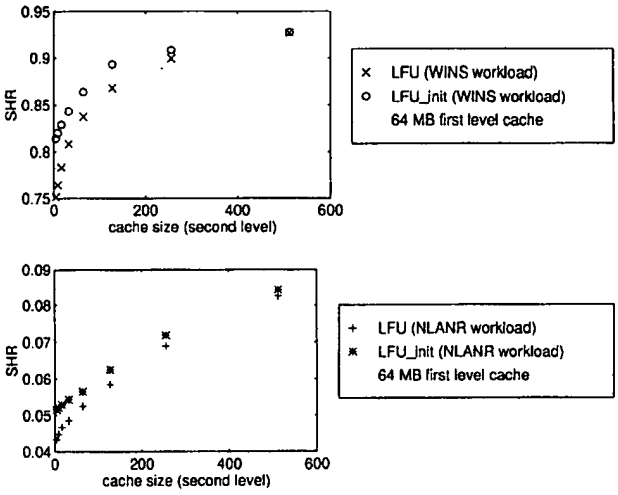


Figure 2. The document hit rates

Using the NLANR workload, where there is plenty of *one-timer* documents, does not impact dramatically the web cache performances (Figures 2 and 3). This confirms the fact that LFU strategy is less sensitive to the sorting process initialization. The *one-timer* documents are removed first in both the LFU and the LFU_init. However, the sorting process initialization has allowed the *one-timer* to remain in the cache for a longer period of time.

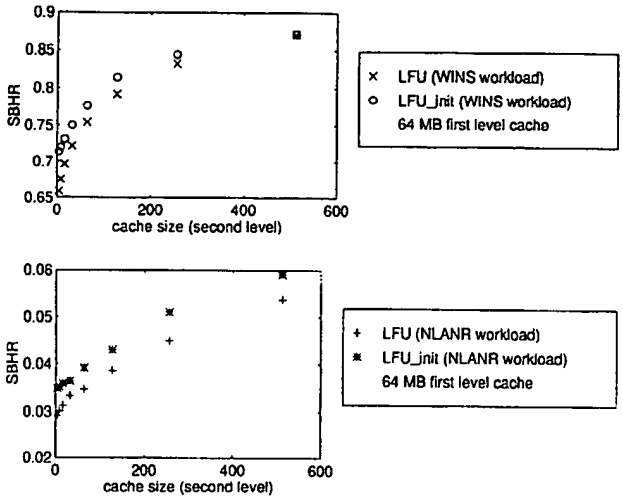


Figure 3. The byte hit rates

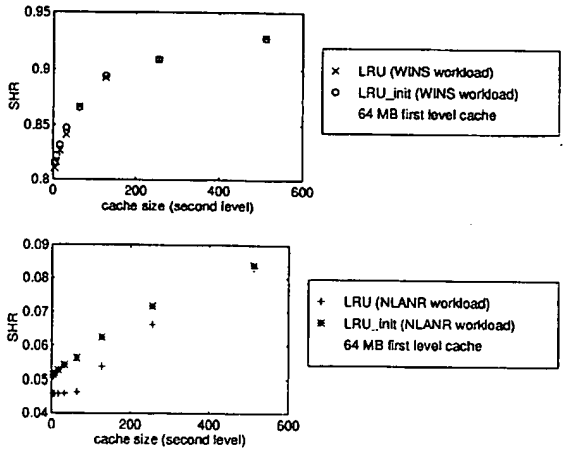


Figure 4. The document hit rates

3.3. The LRU strategy

In this experiment, we have compared the different hit rates obtained when using the LRU strategy with their equivalent rates when we introduce the sorting process initialization (LRU_init). The combination of the LRU strategy and the sorting process initialization, which is also sorting *one-timer* documents according to their entry time, has lead to equivalent global cache performances. Figure 4 shows almost no improvements in both the SHR and SBHR when using the WINS workload. This is probably due to the strong locality of reference, which reduces the number of *one-timer* documents.

When using the NLANR workload which present a low locality of reference, we have first noticed that the second level cache does not have any impact on both the SHR and the SBHR unless it is larger than the first level cache (more details are given in [3]). This phenomenon is the result of the large number of misses recorded in the first level cache, leading to an equivalent number of forwarding documents to the second level cache. The document removal process start being used almost at the same time in the two cache levels, since the same removal policy is used in both of the two cache levels, the probability of a hit in the second level cache is very low. This equivalence between the two cache levels disappears as soon as the second level cache becomes larger, which increases the number of hit in the second level cache.

Combining the LRU strategy with the sorting initialization process, when a large number of *one-timer* documents exist, has increased the web cache performances (SHR and SBHR). However, this combination failed to provide any improvement if the first cache level is relatively small (less than 64 MB). For these cache configurations, the LRU_init did not reduce the high traffic between the two cache levels, which has lead to an intensive use of the document removal process, this has reduce the document storage time in the second level cache. Since the workload do not exhibit a strong reference locality, the probability of a hit in the second level cache is considerably reduced [3].

comparing to the results of LFU strategy presented in the previous section, the impact of the sorting initialization process on both the document hit rate and the byte hit rate is less important when the number of one-timer documents is small, as it is the case of the WINS workload. It seems that the LRU strategy deals better with these documents, it allows them to remain for a longer period of time in the cache, but since their number is reduced they do not disturb the cache performances. The necessary time requires to judge, if a newly requested document is a one-timer document, is satisfied which allows the LRU to distinguish between the two categories of documents. When the number of one-timer documents increases the impact of the initialization process becomes more important, which implies that the transition time of these documents in the cache is large enough to reduces the performance of the system. In this case the combination of the sorting initialization process and the LRU has reduced this transition time which improves the web cache performances.

3.4. The size based strategy

The following experiment shows the impact of the sorting process initialization when using the size of the document as primary sorting key. As it was stated by Williams

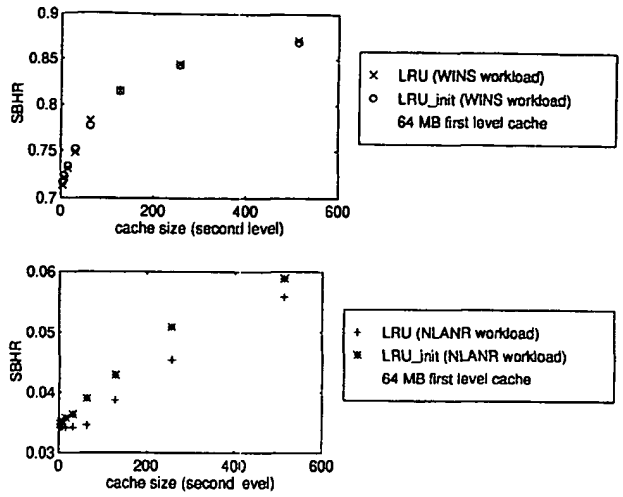


Figure 5. The byte hit rates

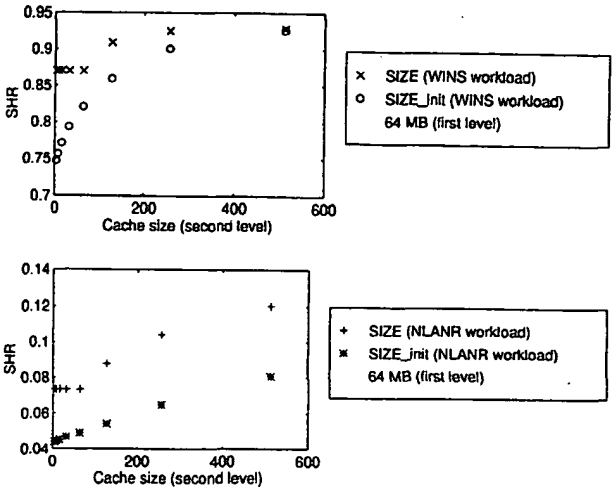


Figure 6. The document hit rates

in [9], the SIZE-based document removal strategy outperformed the LRU and LFU. However, as long as the second level cache is smaller than the first level, it seems that the second level cache does not have a great effect on both the document and bytes rates as shown in [3]. The main reason for this behavior is the fact that SIZE-based policies lead to high hit rates when they remove large size document first. According to this strategy, large size documents are moved to the second level cache which is not necessarily the good way to increase the second level cache hit rates, since the web traffic analysis showed that small size documents are the most frequently referenced.

Clearly, no improvements have been recorded for the document hit rate (SHR) when the sorting process initialization has been introduced as shown in Figure 6, instead a decrease of the document hit rate was recorded when the small cache size configurations are considered. By introducing the sorting process initialization, the *one-timer* documents, which are removed first, are sorted according the time instead of the size. For small cache configuration size, where there is not plenty of memory space, removing old *one-timer* documents instead of the largest one has increased the number of removed documents which reduced the hit rate.

When looking at the byte metric SBHR, we can see the reverse effect when introducing the sorting process initialization. Obviously the SIZE replacement strategy has a bad impact on the byte rates, since it start removing large size document first which increases the number of bytes to be fetch on the next large size document miss. The SIZE_init has reduced the impact of this removal process by introducing the time as a primary removal key within the *one-timer* documents category, this combination between time and size has lead to an improvement in the byte rates as it is shown in Figure 7.

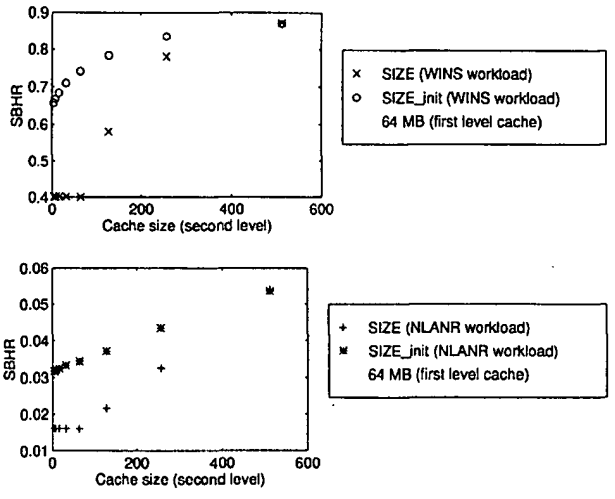


Figure 7. The byte hit rates

The size-based strategy leads to reverse impacts on web cache performances. On one hand it improves the document hit rate on the other hand it reduced the byte hit rates. This result hold true for the two categories of workloads, which means that the number of one-timer documents is not behind this particular behavior. The main reason is the large variations in the size of the cached documents which start

form few kilobytes to few megabytes, in this case removing one large document could make room for thousand of small ones. Not surprisingly, this characteristic leads to the highest document hit rate, since small document are suppose to be the most popular. The bad impact recoded for the byte hit rate is also the result of the large variations of the size, a miss on one large document is equivalent of thousand of misses on small ones in terms of transfered bytes. The sorting initialization process acts as a smoothing parameter which allows to make a tradeoff between the two cache performances.

4. Conclusions

We have investigated the problem of *one-timer* documents and the experiments we have presented in this paper showed the impact of these documents on web cache performance. To deal with *one-timer* documents, we have proposed a new mechanism that assigns a specific priority to the *one-timer* documents, which allows us to constantly keep them at the top of the removal document list. This mechanism was combined with the different document removal policies, and has showed different impacts on the web cache performances. Handling *one-timer* documents separately from other documents does not always lead to better performances, as was shown in the experiments: some replacement strategies such as the LFU intuitively solve the problem of *one-timer* documents and thus a small improvement has been recorded.

The choice of two workloads which include a large (respectively small) number of *one-timer* documents shows the impact of our new mechanism in two extreme situations. It is obvious that the more the workload includes *one-timer* documents the higher the impact of the proposed technique.

For document replacement strategies that combine several parameters according to a well defined mathematical model, such as the Bolot-Hoschka or the NNC, the sorting process initialization badly effects these strategies; it seems that balance created by the mathematical models is disturbed by the partition of the workload in two categories of documents. Another method would be to modify the mathematical models such that they take into account *one-timer* documents rather than imposing a separate process that interferes with these models.

Acknowledgments

The authors are grateful to Henk Muller, Andy Pimentel and Arjan Peddemors for their comments on this paper. The work presented here is part of the JERA project funded by the Dutch HPCN foundation.

References

[1] G. Abdulla and et al. Www proxy traffic characterization with application to caching. *Technical Report, Computer Science Department Virginia Technology*, (CS-97-03):1–20, March 1998.

[2] F. M. Arlitt and C. L. Williamsom. Trace driven simulation of document caching strategies for internet web servers. *Simulations*, 68(1):23–33, January 1997.

[3] A. Belloum and L. Hertzberger. Simulation of a two level cache serve. *Technical Report, Computer Science Department of the Univesity of Amsterdam*, (CS-98-01):1–44, January 1998.

[4] A. Bestavros. Demand-based document dissemination to reduce the traffic and balance load in distributed information system. *In proceeding of the 7th IEEE Symposium on Parallel and Distributed Processing*, October 1995.

[5] H. Braun and K. Claffy. Web traffic characterisation: an assessment of the impact of caching documents from ncsa's web. *WWW:http://www.sdsc.edu/0/sdsc/Research/ANR/.*

[6] J. Gwertzman and M. Seltzer. The case for geographical push-caching. *In proceedings of the Workshop on Hot Operating Systems.*, January 1995.

[7] J. Gwertzman and M. Seltzer. World-wide web cache consistency. *In proceedings of the Usenix technical conference*, January 1996.

[8] M. Gwertzman, J. Seltzer. An analysis of geographical push caching. *http://www.eecs.harvard.edu/.*

[9] S. a. e. a. Williams. Removal politics in network caches for world-wide web documents. *In proceedings of the ACMSIGCOMM*, August 1997.

[10] R. P. Wooste. Optimizing response time, rather than hit rates, of www proxy caches,. *Master Thesis, Blacksburg, Virginia*, August 1996.

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	1	"6449695".pn.	US-PGPUB; USPAT; JPO; IBM_TDB	OR	OFF	2006/06/07 08:22
L2	0	(delet$4 or remov$4 or expir$4 or eliminat$4) with "LRU" with online	US-PGPUB; USPAT; JPO; IBM_TDB	OR	OFF	2006/06/07 08:24
L3	1	(delet$4 or remov$4 or expir$4 or eliminat$4) with "LRU" with offline	US-PGPUB; USPAT; JPO; IBM_TDB	OR	OFF	2006/06/07 08:28
L4	321	online and offline and (cache with updat$4)	US-PGPUB; USPAT; JPO; IBM_TDB	OR	OFF	2006/06/07 09:22
L5	0	l4 and ((delet$4 or remov$4 or eliminat$4 or destroy$4) with "LRU" with original with server with cache with online)	US-PGPUB; USPAT; JPO; IBM_TDB	OR	OFF	2006/06/07 09:24
L6	0	l4 and ((delet$4 or remov$4 or eliminat$4 or destroy$4) with "LRU" with original with server with cache)	US-PGPUB; USPAT; JPO; IBM_TDB	OR	OFF	2006/06/07 09:24
L7	0	l4 and ((delet$4 or remov$4 or eliminat$4 or destroy$4) with "LRU" with original with server)	US-PGPUB; USPAT; JPO; IBM_TDB	OR	OFF	2006/06/07 09:25
L8	14	l4 and ((delet$4 or remov$4 or eliminat$4 or destroy$4) with "LRU" with cache)	US-PGPUB; USPAT; JPO; IBM_TDB	OR	OFF	2006/06/07 09:50
L9	0	l4 and ((delet$4 or remov$4 or eliminat$4 or destroy$4) with "LRU" with cache with updat$4)	US-PGPUB; USPAT; JPO; IBM_TDB	OR	OFF	2006/06/07 09:50
L10	9	l4 and ((delet$4 or remov$4 or eliminat$4 or destroy$4) with ("LRU" or time) with cache with updat$4)	US-PGPUB; USPAT; JPO; IBM_TDB	OR	OFF	2006/06/07 10:55
L11	0	l4 and ((delet$4 or remov$4 or eliminat$4 or destroy$4) with ("LRU" or time) with cache with updat$4 with online)	US-PGPUB; USPAT; JPO; IBM_TDB	OR	OFF	2006/06/07 10:55

EAST Search History

L12	0	l4 and ((delet$4 or remov$4 or eliminat$4 or destroy$4) with ("LRU" or time) with cache with updat$4 with (online or backup))	US-PGPUB; USPAT; JPO; IBM_TDB	OR	OFF	2006/06/07 10:56